Natural Language Engineering (2024), **30**, pp. 132–149
doi:10.1017/S1351324923000049

ARTICLE

An unsupervised perplexity-based method for boilerplate removal

Marcos Fernández-Pichel*[†], Manuel Prada-Corral[†], David E. Losada, Juan C. Pichel and Pablo Gamallo

Centro Singular de Investigación en Tecnoloxas Intelixentes (CiTIUS), Universidade de Santiago de Compostela, 15782 Santiago de Compostela, Galicia (Spain)
*Corresponding author. E-mail: marcosfernandez.pichel@usc.es

(Received 28 April 2022; revised 9 January 2023; accepted 17 January 2023; first published online 21 February 2023)

Abstract
The availability of large web-based corpora has led to significant advances in a wide range of technologies, including massive retrieval systems or deep neural networks. However, leveraging this data is challenging, since web content is plagued by the so-called boilerplate: ads, incomplete or noisy text and rests of the navigation structure, such as menus or navigation bars. In this work, we present a novel and efficient approach to extract useful and well-formed content from web-scraped data. Our approach takes advantage of Language Models and their implicit knowledge about correctly formed text, and we demonstrate here that perplexity is a valuable artefact that can contribute in terms of effectiveness and efficiency. As a matter of fact, the removal of noisy parts leads to lighter AI or search solutions that are effective and entail important reductions in resources spent. We exemplify here the usefulness of our method with two downstream tasks, search and classification, and a cleaning task. We also provide a Python package with pre-trained models and a web demo demonstrating the capabilities of our approach.

Keywords: Perplexity; Boilerplate removal; Information Retrieval; Text classification; Text Pre-processing

1. Introduction

Web crawling is a widely used technique to obtain textual collections for multiple purposes in the fields of Natural Language Processing (NLP) and Information Retrieval (IR). However, isolating relevant content from web pages is far from trivial. Noisy links, menus, navigation bars and other extraneous elements – the so-called *boilerplate* – impoverish the performance of downstream applications (Kohlschütter 2009; Yi *et al.* 2003) and impose a substantial load on the systems (e.g., leading to higher storage requirements or larger search and classification times).

Traditional approaches to this problem follow rule-based or supervised learning approaches (Vieira *et al.* 2006; Vogels *et al.* 2018). In this paper, we propose a novel unsupervised strategy to perform the removal of boilerplate content. A Language Model (LM) is used to compute a perplexity value, which computes the logarithm of the probability for a given sentence according to the model distribution. This score can be seen as the 'unlikeliness' of the sentence, which can, in turn, be employed as a way to identify boilerplate and other non-language noise.

Our proposal also addresses efficiency issues by reducing execution times and storage requirements. As a non-supervised method, no specific training process is required and the resulting

[†] These authors contributed equally to this work.

systems are lighter, computationally efficient, but still effective. This is intimately related to the current trend on Green AI solutions. Some authors (Duhart *et al.* 2019; Henderson *et al.* 2020; Schwartz *et al.* 2020) have begun to warn about the problem of focusing only on the performance metrics of algorithms and ignoring other economic, environmental and social costs. Schwartz *et al.* (2020) also underlined the need to guarantee access to state-of-the-art technologies without huge investments being made. They propose to increase the importance of efficiency criteria when evaluating AI algorithms and studies. We believe that perplexity can become a valuable pre-processing element with potential to support multiple text mining applications, and, more specifically, it can act as a formal pruning device that leads to energy efficient solutions.

The main contributions of this paper can be summarised as follows:

- A novel unsupervised method for boilerplate removal based on Language Models and perplexity estimation is proposed.
- In terms of effectiveness, we show that this perplexity-based technique leads to improvements in an information retrieval search task and in a classification task. To that end, we have experimented with different datasets and evaluation metrics. In addition, we also demonstrate the potential of perplexity for a well-established webpage cleaning task.
- Efficiency criteria have been carefully considered, and we show that removing boilerplate following a perplexity approach produces substantial reductions in storage space and yields low execution times. The offline (pre-processing) requirements of the boilerplate removal stage are quickly compensated for the computational advantages associated with the resulting noise-free web contents when they are used in search or classification tasks.
- A Python package named PyPlexity has been built, and it is freely available to use[a]. It offers all the functionalities developed for this experimentation: removing HTML tags from a web document and computing perplexity for a given sentence, file or dataset. An online demo[b] is also available to test our models.

The rest of the paper is organised as follows. Section 2 reviews some papers related to our research. In Section 3, we explain our unsupervised method for boilerplate removal. Section 4 reviews the experimental settings utilised to conduct the study. Performance and efficiency results for all tasks are reported in Section 5. Section 6 discusses the main findings, and Section 7 presents the Python package and web application. The paper ends with some conclusions.

2. Related work

Seminal approaches to boilerplate removal were rule-based. For example, Finn *et al.* (2001) utilised the position of HTML tags in web pages to determine a series of heuristics for extracting the main content. However, this method is rather rigid and cannot handle web documents that follow new structural patterns.

Later, some research teams considered DOM trees and HTML pages divided into blocks and employed supervised learning methods to discern between useful and non-useful blocks. Bauer *et al.* (2007) used Support Vector Machines (SVMs) to classify blocks based on linguistic, visual and structural features. Spousta *et al.* (2008) proposed a cleaning tool that follows a sequence-labelling approach based on Conditional Random Fields. Kohlschütter *et al.* (2010) performed a comparison between shallow textual features and more sophisticated approaches. This comparative study was run with SVMs and under a block-oriented strategy. Pomikálek

[a]https://github.com/citiususc/pyplexity
[b]https://tec.citius.usc.es/pyplexity/

(2011) also presented an unsupervised approach (named *jusText*) to deal with boilerplate based on hyperparameter exploration. This was the first algorithm that took context into account when identifying blocks of text.

A recent proposal is Web2Text (Vogels *et al.* 2018), which introduces a set of features from adjacent neighbours in the DOM tree and utilises deep learning techniques to predict each block's category. Leonhardt and colleagues (Leonhardt *et al.* 2020) noticed that these models require a large number of hand-crafted features and annotated training data. For this reason, they offered an alternative model that does not require pre-processing and directly classifies sequences of raw HTML from few training examples.

Most of these studies have shown the usefulness of their methods in tasks such as IR search, where pre-processing had been classically confined to simple strategies such as stemming or stop-word removal (Kannan *et al.* 2014). In this study, we try to go one step further by evaluating the effectiveness and efficiency of our boilerplate removal method not only for a search task but also for document classification and cleaning tasks.

The main novelty of our study lies in the utilisation of perplexity as an indicator of well-formed text. Thus, our method can not only remove unnecessary scraped HTML blocks, but also malformed content. Instead of block segmentation, we propose a more general sentence segmentation technique. We build a Language Model and employ perplexity to estimate sentence likelihood (see Section 3 for more details). Related to our work, Wenzek *et al.* (2019) employed perplexity as a proxy of quality of documents and scored documents to create curated monolingual corpora. We are not interested in removing documents from the collections but, rather, we define a document pre-processing technique able to remove noisy parts of the original texts.

Other uses of perplexity-based metrics have been suggested for some language-related tasks, such as tweet classification (González 2015), language distance (Gamallo *et al.* 2017; Campos *et al.* 2020) or misinformation identification (Lee *et al.* 2020). Wu *et al.* (2020) also employed perplexity scoring as an evaluation measure to estimate the quality of their entity extraction model. Solorio *et al.* (2011) used NLP techniques to determine linguistic profiles in children, also using an LM background and the role of perplexity as an assessment metric.

As a final note, observe that we adopted in our study perplexity as our main indicator device, but other language metrics could have been considered. For example, multiple language divergence measures, such as Kullback-Leibler divergence (KLD) (Csiszár 1975), could be applied to this task. The exploration of other divergence-based metrics and the study of their connection with the perplexity-based approach reported here is left for future work.

3. Methodology

Perplexity is a measure that has been mainly employed to evaluate LMs without targeting a specific downstream task (Sennrich 2012) (i.e., as an intrinsic evaluation of models of language). A perplexity model indicates how well the data fit into the model distribution. If we assume that the model distribution is correct and unbiased, perplexity allows us to identify noisy data and outliers. In our case, we adopt perplexity as an indicator of potential boilerplate within a webpage.

The perplexity scores represents a metric about how well a language model fits a text sample, for instance a word or a sentence. Low perplexity suggests that the language model is good at predicting a given word, sentence or textual extract, while high perplexity indicates that the language model is not good for that prediction. More formally, the perplexity (called *Perpl* for short) of a language model on a text sample (e.g., a sentence) is the exponential of the cross entropy of the given text. Given a sentence $S = w_1, w_2, \dots, w_n$ and a language model LM with n-gram probabilities, $P()$, estimated on a large corpus, the *Perpl* of S given the n-gram model LM is computed as follows:

$$Perpl(S, LM) = 2^{-\frac{1}{n}\cdot\sum_i^n \log_2 P(w_i|w_1^{i-1})} \quad (1)$$

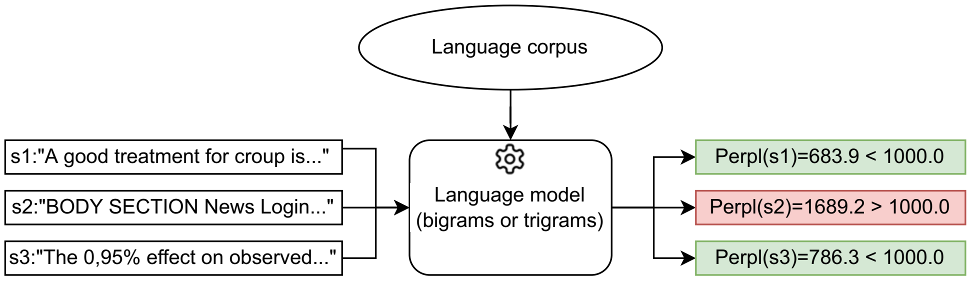

Language corpus
s1:"A good treatment for croup is..."
s2:"BODY SECTION News Login..."
s3:"The 0,95% effect on observed..."
Language model (bigrams or trigrams)
Perpl(s1)=683.9 < 1000.0
Perpl(s2)=1689.2 > 1000.0
Perpl(s3)=786.3 < 1000.0

Figure 1. Perplexity model for boilerplate removal.

where n-gram probabilities $P(\cdot)$ of a word in position i given the immediate sequence to the left are defined as:

$$P(w_i|w_1^{i-1}) = \frac{C(w_1^{i-1}w_i)}{C(w_1^{i-1})} \tag{2}$$

Equation (2) estimates the n-gram probability by dividing the observed frequency (C) of a particular sequence of words by the observed frequency of the same sequence without the last word.

The use of perplexity for cleaning and pruning a corpus is illustrated in Figure 1. In our method, perplexity is computed on all input sentences and those with *Perpl* scores above a certain threshold (e.g., > 1000) are removed. This represents a simple but potentially effective way to remove noisy parts of webpages. Traditional pruning mechanisms (e.g., word-level techniques such as stopword removal, frequency-based word pruning or stemming and lemmatisation) are also rather straightforward but, over the years, have become standard elements in multiple text mining and retrieval tools. Note also that the removal of noisy and non-relevant contents can be beneficial not only in terms of effectiveness but also in terms of efficiency (e.g., lighter indexes or data structures, faster access times).

3.1 Perplexity models

Two different datasets were selected to build the Language Models: the CORD-19 dataset (Wang *et al.* 2020) and the British National Corpus (BNC) (Consortium *et al.* 2007). The first is a language corpus associated with a specific topic, while the second represents a more general use of language. CORD-19 contains 50K papers with over 41K full texts about COVID-19 and related historical coronaviruses such as MERS or SARS. We employed the abstracts of the papers to build the LM since abstracts are well-formed and contain succinct sentences. These abstracts add up 510 MB of text. BNC is a 100 million word collection of written and spoken British English collected from several sources and created by Oxford University. Work on the corpus began in 1991 and the written part, which is the one that interests us the most here, represents 90% of the total, including extracts from newspapers, specialist journals, academic books, and popular fiction. In total, we worked with 620 MB of data to build the LM.

We preferred highly curated texts to create the LMs, and we opted for two collections from different domains and genres, one more specific and another more general. This helps to study the influence of specificity vs generality on perplexity-based pruning of webpages. For this task, there was no need for an intricate model, since the objective is to just discern between well-formed

Table 1. General statistics of search collections

Task	Number of queries	Query ids	Avg rel per query	Used for	Doc collection	Number of docs	Type of docs
TREC 2019 Decision Track	51	1–51	82	Hyperparameter selection	ClueWeb12-B13	50M	webpages
TREC 2013 Web Track	50	201–250	88	Test	ClueWeb12-B13	50M	webpages

sentences and notorious boilerplate. We chose simple bigrams and trigrams probabilistic models, which are both fast and adequate for perplexity computation.

4. Experimental settings

4.1 Datasets

The perplexity models built from CORD-19 and BNC were evaluated with two test collections designed for specific tasks (see Table 1 for details). The ClueWeb12-B13[c] collection was employed to perform experiments of indexing and search. This collection contains approximately 50 million pages. We ran tests with two different sets of queries and relevance assessments (one set of search topics – from the medical domain – obtained from the TREC 2019 Decision Track (Abualsaud *et al.* 2019) and another set of general topics from the TREC 2013 Web Track (Collins-Thompson *et al.* 2013)). The first set was used to determine the only parameter that our method requires: the maximum admissible value of perplexity per sentence. This parameter selection was then validated with the second set of test queries. The results are reported in Section 5.

A second class of experiments was executed to assess the effectiveness of the perplexity-based approach within a text classification task (webpage classification). To that end, the WebKB collection[d] was chosen. It contains 8282 web pages from four different universities classified into seven different categories (courses, departments, faculty, projects, staff, students and other).

Finally, a third set of experiments was conducted to test the ability of our solution to clean up web content in the context of a well-known competition for cleaning webpages. More specifically, the shared-task CleanEval (Baroni *et al.* 2008) provided the ideal framework for these last experiments. The CleanEval dataset contains a random sample of web corpora which was collected by making queries to Google (only html pages were collected). We employed 625 CleanEval webpages written in English. The collection contains the original webpages and webpages cleaned by recruiting human annotators who read the webpages and removed noisy content.

4.2 Metrics

For testing retrieval performance, the following metrics were used:

- Precision at n ($P@n$): represents the fraction of the top n documents of the ranking that are relevant to the user's information need (Croft *et al.* 2010):.

$$P@n = \frac{\sum_{p=1}^{n} rel(p)}{n} \tag{3}$$

where $rel(p)$ equals 1 if the item at position p is a relevant document and 0 otherwise.

[c]http://lemurproject.org/clueweb12/
[d]http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

- Average Precision (AP): it summarises the ranking by averaging the precision scores at the rank positions where a relevant document is founde[e] (Croft et al. 2010). In Equation (4), rels is the total number of relevant documents and *P@n* is the precision at a cut-off n. The mean AP (MAP) for a set of queries is the mean of average precision scores for each query. It is a popular method to assess the effectiveness of a ranked set of results.

$$AP = \frac{1}{rels} \cdot \sum_{n=1}^{rels} P@n \cdot rel(n) \tag{4}$$

- Normalised Discounted Cumulative Gain at n (*NDCG@n*): NDCG measures the quality of search results taking into account different grades or levels of relevance (Croft et al. 2010):

$$NDCG = \frac{DCG}{IDCG} \tag{5}$$

$$DCG = \sum_{p=1}^{n} \frac{2^{\mathrm{rel}}_{p} - 1}{log_2(p+1)} \tag{6}$$

  DCG, Discounted Cumulative Gain (Equation (6)), defines the user's gain as a measure that grows as he/she goes from the top of the ranking to lower positions. Under *NDCG*, the gain produced by each ranked document depends on its position. Gains obtained by relevant documents at higher positions are greater than those from relevant documents at lower ones. For example, a highly relevant document at a low-rank position is substantially penalised. To that end, each gain is penalised by a discounting factor ($log_2(p+1)$). The *DCG* values are normalised by dividing the *DCG* scores by the ideal *DCG* (*IDCG*, which represents the gains obtained by an oracle system that ranks documents by decreasing order of their actual relevance). The *NDCG@n* score represents the accumulated gain that the user obtained from examining the top *n* results.

All these metrics are computed for each available query and the reported figures represent the mean value across all queries. To analyse the statistical significance of the performance difference between two results, we employ the Wilcoxon test on the paired values (one from each query). Parapar *et al.* (2021, 2020) showed that the Wilcoxon test is a highly reliable test to compare retrieval systems (yields more statistical power and fewer type I errors). The significance tests help to determine whether or not each observed difference is anecdotal.

For evaluating classification performance, the F1 score (harmonic mean between precision and recall) for each class and macro average F1 (unweighted mean of F1 per class) are reported.

In this study, we also aim at improving computational efficiency, and, thus, we report here some time measures, storage improvements and an estimate of carbon savings (Lacoste *et al.* 2019). The elimination of noisy document's parts has a potential to reduce time or space complexity, and it is important to quantify these improvements.

For the CleanEval shared task, the effectiveness metric measures the similarity between a cleaned version of the file (produced by a given cleaning algorithm, for example, our perplexity-based method) and the gold standard (produced from human annotators). To that end, the task considers *Levenshtein edit distance* as the main scoring method. It computes the distance between two strings given the fewest operations required to transform one into the other. The final measure is the percentage of misalignment between files (in our case, considering only text; HTML labels are skipped).

[e]and relevant documents that were not retrieved contribute with a 0 to the AP score (i.e. their *P@n* is set to 0).

4.3 Search and classification models

For the retrieval experiments, the collection was indexed using the Anserini search engine (Yang *et al.* 2018), and the retrieval model utilised was the query likelihood model (QL) (Ponte and Croft 2017). This is a well-known probabilistic retrieval approach that assumes that the query is generated by sampling words from the document and that each query term is independent, meaning that the probability of the query is a product of the probability of each term. Equation (7) presents the QL document relevance score, where $p(q|d)$ is the conditional probability of the query q given the document d. To account for query terms that do not appear in the document, probability values need to be smoothed using a reference language model or corpus. In our experiments, we utilised Dirichlet Smoothing (Equation (8)). A full description of probabilistic language models for IR, which shows the advantages of QL with Dirichlet smoothing and its solid length normalisation abilities, is available at Losada and Azzopardi (2008).

$$\text{QL\_score}(q, d) = \log p(q|d) \tag{7}$$

$$\text{QL\_score}_{\text{DIR}}(q, d) = \sum_{w \in q,d} c(w, q) \cdot \log\left(1 + \frac{c(w, d)}{\mu \cdot p(w|C)}\right) + |q| \cdot \log \frac{\mu}{\mu + |d|} \tag{8}$$

$c(w, q)$ ($c(w, d)$) is the number of occurrences of word w in the query (document). $p(w|C)$ is the probability of the word in the background corpus, and $|q|$ and $|d|$ are the lengths of the query and document, respectively. μ is the smoothing parameter, which was set to 2000 in our experiments.

For document classification, state-of-the-art Transformers models were used (Vaswani *et al.* 2017). More specifically, we evaluated and compared BERT (Devlin *et al.* 2018) and RoBERTa (Liu *et al.* 2019) models. These models consist only of an encoder architecture and have proven ability in outperforming traditional classifiers. We experimented with the base and large configurations of both models. To that end, Ernie[f], the open-source library, was utilised.

Following Sun *et al.* (2002), we did cross-validation by always leaving the webpages from one of the universities out. The classification collection is highly imbalanced, and, thus, we followed standard practise to set the error weights from the proportion of cases in each class.

4.4 Hardware platform

Different hardware configurations were used, depending on the task. The tag removal and perplexity computation for the ClueWeb dataset (which corresponds to *TREC 2019 Decision Track* and *TREC 2013 Web Track* tasks) was carried out in a Big Data computing cluster. This cluster is formed by 15 nodes, each one with two Intel Xeon E5-2630 v4 and 384GB of RAM, for a total of 20 cores per node. The computing load for these tasks was distributed across the 300 available cores using data parallelism. Once the tag removal and perplexity computation for the dataset was completed, indexing and search (supported by Anserini) were run in one node of the cluster.

The classification task required a GPU-powered server. For this reason, experiments were conducted in a single server with two Intel Xeon Gold 5220, 192GB of RAM and two Nvidia Tesla V100S. These GPUs efficiently support the training processes required by BERT and RoBERTa classification models.

Finally, the cleaning task was computed on a single node with an Intel i7-9700K CPU @ 3.60GHz, 32GB of RAM and no GPU capabilities.

[f]https://github.com/labteral/ernie

5. Results

5.1 Search experiments

5.1.1 TREC 2019 decision track

The first part of the experiments was conducted with queries and relevance assessments from the TREC 2019 Decision Track (DT19). This test collection consists of search topics related to the medical domain and documents crawled from the web (ClueWeb dataset).

The DT19 collection was used for optimising the only hyperparameter of the boilerplate removal method: the perplexity cut-off value. The higher the perplexity cut-of,f the fewer sentences are removed. We experimented with the following perplexity cut-off values: 1K, 2K, 4K, 6K, 8K, 10K, 12K and 14K.

Table 2 shows the results by comparing the effectiveness of different levels of removal against the retrieval baseline, which does not remove any sentence. Most of the tested models and thresholds outperform the baseline and in some cases the improvements are statistically significant. The improvements are solid for high-precision metrics (P@5 and P@10) but also for AP and NDCG, which take into account the entire ranking of documents. Only a couple of instances lead to performance decreases that are statistically significant (and this only happens for AP). This is reasonable since AP is a recall-oriented measure. Our cleaning technique eliminates malformed text but the method is not perfect, and, thus, some pruned sentences might be actually relevant. This affects AP in some of the tested configurations. In contrast, the removal method proves to be powerful for the three precision-oriented metrics (P@5, P@10 and NDCG@10).

This exploration of the perplexity cut-off suggests that we can safely find a cut-off configuration that is robust and works well for precision-oriented and recall-oriented metrics. The reader should also bear in mind that it is important to keep (or even improve) retrieval effectiveness, but space savings and other efficiency-oriented factors are other important aspects associated to perplexity-based removal (this will be further discussed in Section 5.1.3).

Not surprisingly, BNC-based models outperform CORD-19 ones. This confirms that the BNC-based approach, which removes noisy sentences based on a general model of language, is more apt to determine which sentences should remain. The DT19 search topics are medical queries and, thus, somehow close to CORD-19 but, still, this more focused language data does not give any added value. Observe that it is convenient that the approach does not require a topic-specific corpus to build the perplexity models. A general language corpus suffices to filter out noisy and off-topic contents, and the BNC-based method would be applicable across general-purpose search tasks and collections.

Regarding bigrams vs trigrams, the latter models yielded slightly better effectiveness. Taking these results into account, we adopted the following models, which constitute a good balance among all the performance metrics: BNC trigram models (8K and 10K), and BNC bigram models (8K and 10K). These variants, which are marked in light grey colour in the table, are not always the best performers but they represent a solid configuration that often leads to statistical significant improvements.

5.1.2 TREC 2013 web track

Next, the selected models were evaluated with another test collection, the TREC 2013 Web Track (WT2013). This dataset has also a large corpus of webpages crawled from the web (ClueWeb12 dataset) and includes search topics that represent general information needs (informational or navigational queries).

The results of this experimentation are shown in Table 3. In most of the cases, the perplexity-based variants lead to higher effectiveness, and the method only yields minor decreases in AP. Note also that AP is not a crucial metric for most web retrieval tasks, where high recall is rarely pursued. This outcome further reinforces the potential of the approach not only to produce lighter and less noisy indexed data but also to maintain and even improve retrieval performance.

Table 2. TREC 2019 Decision Track. Effect of different perplexity cut-off values (reported in brackets) on retrieval performance. The ↑/↓ symbols indicate whether the method significantly improves or not (Wilcoxon test, $\alpha = 0.05$) over the baseline (no perplexity-based removal)

	P@5	P@10	AP	NDCG@10	Size (GB)
BASELINE	0.564	0.546	0.328	0.458	382
Bigrams model					
CORD-19 (1K)	0.592	0.540	0.321	0.484	51
CORD-19 (2K)	0.592	0.550	0.332	0.479	61
CORD-19 (4K)	0.608	0.564	0.335	0.489	72
CORD-19 (6K)	0.596	0.566	0.335	0.489	78
CORD-19 (8K)	0.584	0.556	0.326	0.477	82
CORD-19 (10K)	0.584	0.552	0.325	0.468	85
CORD-19 (12K)	0.580	0.562	0.323	0.473	88
CORD-19 (14K)	0.588	0.568	0.326	0.477	90
BNC (1K)	0.620	0.554	0.314	0.484	60
BNC (2K)	0.608	0.580	0.333	0.492	69
BNC (4K)	0.604	0.578	0.338	0.487	78
BNC (6K)	0.620	0.578	0.337	0.487↑	89
BNC (8K)	0.600	0.568	0.334	0.481	93
BNC (10K)	0.600	0.576	0.333	0.486↑	96
BNC (12K)	0.604	0.576	0.333	0.486	99
BNC (14K)	0.600	0.566	0.335	0.481	101
Trigrams model					
CORD-19 (1K)	0.564	0.526	0.293↓	0.455	43
CORD-19 (2K)	0.612	0.552	0.328	0.485	55
CORD-19 (4K)	0.604	0.576	0.336	0.493	66
CORD-19 (6K)	0.604↑	0.564	0.334	0.488	72
CORD-19 (8K)	0.608↑	0.564	0.338	0.487	77
CORD-19 (10K)	0.608	0.572	0.336	0.490↑	80
CORD-19 (12K)	0.592	0.572	0.332	0.485	83
CORD-19 (14K)	0.596	0.566	0.330	0.480	86
BNC (1K)	0.584	0.546	0.293↓	0.455	52
BNC (2K)	0.628 ↑	0.560	0.324	0.490	63
BNC (4K)	0.620↑	0.582	0.335	0.494	72

Table 2. Continued.

	P@5	P@10	AP	NDCG@10	Size (GB)
BNC (6K)	0.620↑	0.580	0.338	0.491	77
BNC (8K)	0.624↑	0.576	0.341	0.486	81
BNC (10K)	0.620↑	0.578	0.340	0.490↑	84
BNC (12K)	0.604	0.580	0.339	0.491↑	87
BNC (14K)	0.596	0.578	0.335	0.486↑	89

Table 3. TREC 2013 Web Track. The ↑/↓ symbols indicate whether the method significantly improves or not (Wilcoxon test, $\alpha = 0.05$) over the baseline (no perplexity-based removal)

	P@5	P@10	AP	NDCG@10	Size (GB)
BASELINE	0.236	0.230	0.039	0.154	382
Bigrams model					
BNC (8K)	0.268	0.230	0.037	0.163	93
BNC (10K)	0.264	0.248	0.037	0.165	96
Trigrams model					
BNC (8K)	0.264	0.238	0.037	0.171	81
BNC (10K)	0.268	0.230	0.037	0.165	84

Additionally, there is not a noticeable difference between bigram and trigram models. Both alternatives perform roughly the same. This is another interesting outcome since bigram models are simpler and thus less costly.

5.1.3 Efficiency

Improving effectiveness is an attractive feature of the perplexity-based removal but efficiency is also a major dimension that needs to be taken into account. The first notable advantage of our method is space saving. The sizes of the corresponding indexes are reported in Tables 2 and 3 (last column), while Figure 2 plots the sizes in a graphical way (the leftmost bar represents the baseline situation with no perplexity-based removal). Compared with the original collection, we can save up more than 75% of the storage cost of the indexing structures (e.g., 96 GB vs 382 GB for the BNC-10k model).

This space saving also translates into several time improvements. The reduced size of the collection results in a substantial decrease of the computing time required to index and search (see Table 4). Our method requires two cleaning phases before indexing: the removal of HTML tags from the entire collection and the actual computation and removal of perplexity at sentence level. As stated in Section 4.4, both operations were carried out in parallel in a Big Data cluster. However, with multiple users doing search online, a crucial measure for an IR engine is search time and that is where our method stands out. For example, for the BNC bigrams model with a threshold set at 8K, the search of 10,000 queries is approximately 1.95 times faster than the same search against the original index. Indexing and searching were executed on a single node because of the restrictions of the retrieval library utilised. Nonetheless, these results can generalise to an entire cluster.

Table 4. Time measurements (HH:MM:SS) for the baseline and some perplexity-based variants

	Tag cleaning	Perplexity computing	Index building	Search (50 queries)	Search (10K queries)
BASELINE	–	–	03:24:23	00:00:45	125:25:00
Bigrams model					
BNC (8K)	00:45:00	00:45:00	01:01:42	00:00:23	64:11:40
BNC (10K)	00:45:00	00:45:00	01:07:48	00:00:25	70:03:20
Trigrams model					
BNC (8K)	00:45:00	00:45:00	01:00:08	00:00:24	65:31:40
BNC (10K)	00:45:00	00:45:00	01:04:04	00:00:25	68:56:40

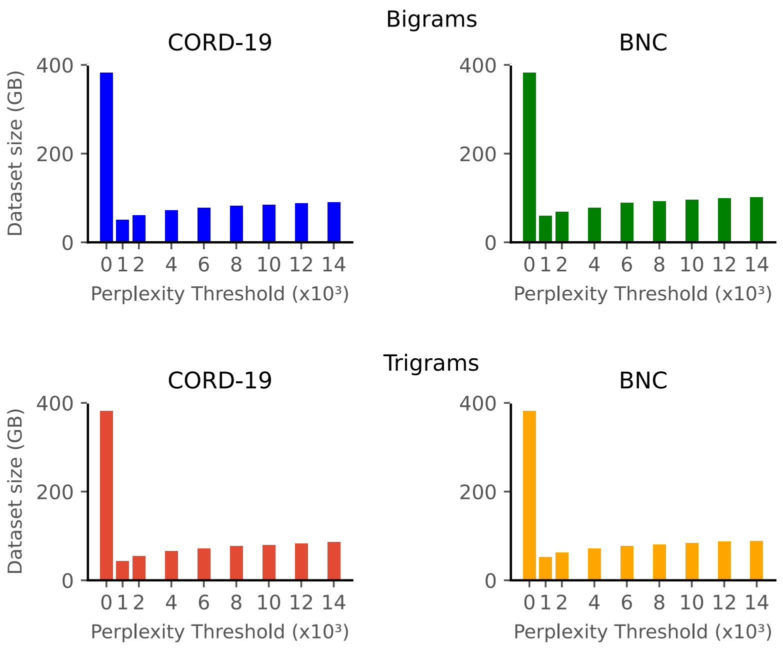


Figure 2. Space savings derived from using our model in the IR search task.

Consequently, this translates into important carbon emission reductions. This reduction grows with the number of searches done against the collection. In Table 5, we show the estimated carbon emissions for our system. Carbon emission were estimated following the methodology presented in Lacoste *et al.* (2019), while carbon intensity data for our region and energy provider was taken from Moro and Lonza (2018). We first calculated the equivalent CPU-hours required for the computation in each step and then derived the estimated carbon emissions with the following relation:

$$\text{eq. kg of CO}_2 = \frac{t \cdot C_e \cdot W_{\text{cpu}}}{1000} \tag{9}$$

Table 5. Carbon emissions in kg of CO_2 for the baseline and some perplexity-based variants

	Tag cleaning	Perplexity computing	Index building	Search (10K queries)	Search (1M queries)
BASELINE	–	–	0.20	0.15	14.54
Bigrams model					
BNC (8K)	0.65	0.65	0.06	0.07	7.44
BNC (10K)	0.65	0.65	0.07	0.08	8.12
Trigrams model					
BNC (8K)	0.65	0.65	0.06	0.08	7.60
BNC (10K)	0.65	0.65	0.06	0.08	7.99

where t is the equivalent CPU-hours of computation (taking parallelisation into account), $C_e = 0,341$ is the carbon efficiency coefficient of the grid (measured in kg CO_2eq/kWh), and W_{cpu} is the Thermal Design Power of the CPU in watts.

With the 8K BNC bigrams model, the answer to 10,000 searches requires approximately half carbon emissions of the baseline. Consider, for example, the case of a real-world IR system. Google reported more than 2 million searches per minute in 2012, its latest statistics available[g]. Taking all the perplexity computing, indexing and inference emissions into account, our method reduces on average 5.58kg of CO_2 per million queries, which translates into more than 22 tons of CO_2 saved each day.

5.2 Classification experiments: WebKB

Let us evaluate now the performance of the perplexity-based models for another text-related challenge, a document classification task. To that goal, we adopted the experimental methodology and collection utilised in Sun *et al.* (2002), but we worked with newer classifiers based on transformers (instead of traditional SVMs).

The results are shown in Table 6. The first row corresponds with the baseline that, in this case, takes the webpages and only removes the HTML tags. This is a standard pre-processing approach in web classification. Our cleaning methods allowed the classifier to perform better in many cases. The bigrams models (and, particularly, the ones with the threshold set to 8K) are the best performers. Another interesting outcome is the low performance obtained by the baseline with the BERT large model. A plausible explanation for this is that BERT large is a model with a huge number of parameters, and it would need a larger amount of training data to generalise better. In any case, the BERT large model also benefits from our pre-processing methods, which only keep useful information and remove boilerplate. This difference is not as noticeable with RoBERTa large, as it is an improved and robust model to avoid such problems.

Overall, these results suggest that a base model with perplexity-based removal of noisy sentences is comparable to (or better than) more sophisticated (and more inefficient) models based on a larger set of parameters. As a matter of fact, the highest F1 macro (0.6350) is obtained by combining the RoBERTa base model with perplexity-based pre-processing (bigrams, 8k threshold).

[g]https://archive.google.com/zeitgeist/2012/#the-world

Table 6. Classification results for WebKB dataset. For each block, the top performer (F1 macro) is bolded

	F1 Macro	F1 Course	F1 Department	F1 Faculty	F1 Project	F1 Staff	F1 Student	F1 Other
BERT base								
BASE. CLEAN	0.5818	0.6765	0.0942	0.7676	0.3914	0.4594	0.8160	0.8678
Bigr. BNC (8K)	**0.5832**	0.6407	0.1083	0.7555	0.4327	0.5425	0.7492	0.8534
Bigr. BNC (10K)	0.5667	0.5592	0.1278	0.7538	0.4123	0.4806	0.7943	0.8391
Trigr. BNC (8K)	0.5128	0.5217	0.0294	0.7064	0.4201	0.3564	0.7221	0.8332
Trigr. BNC (10K)	0.5280	0.5358	0.1377	0.7001	0.3881	0.4058	0.7098	0.8188
RoBERTa base								
BASE. CLEAN	0.6109	0.6948	0.2190	0.7589	0.4183	0.4753	0.8230	0.8875
Bigr. BNC (8K)	**0.6350**	0.6980	0.2917	0.7553	0.5444	0.5425	0.7957	0.8911
Bigr. BNC (10K)	0.6137	0.6781	0.1881	0.7681	0.4422	0.5497	0.7871	0.8828
Trigr. BNC (8K)	0.5928	0.6293	0.2399	0.7523	0.4301	0.4739	0.7493	0.8805
Trigr. BNC (10K)	0.5544	0.5928	0.1210	0.7571	0.4182	0.4555	0.7203	0.8232
BERT large								
BASE. CLEAN	0.4309	0.4749	0.2131	0.5979	0.2332	0.2623	0.6194	0.6155
Bigr. BNC (8K)	**0.5815**	0.6454	0.1372	0.7806	0.4568	0.4111	0.7564	0.8831
Bigr. BNC (10K)	0.5211	0.5313	0.1165	0.7368	0.3600	0.3978	0.7555	0.7502
Trigr. BNC (8K)	0.5419	0.5387	0.1061	0.7444	0.4837	0.3339	0.7445	0.8418
Trigr. BNC (10K)	0.5320	0.5576	0.0583	0.7244	0.3916	0.4340	0.7469	0.8108
RoBERTa large								
BASE. CLEAN	0.6255	0.7406	0.1929	0.7809	0.4148	0.5357	0.8157	0.8980
Bigr. BNC (8K)	0.6169	0.6632	0.1387	0.7490	0.4964	0.5476	0.8227	0.9013
Bigr. BNC (10K)	0.6282	0.7198	0.2137	0.7955	0.4478	0.4701	0.8416	0.9050
Trigr. BNC (8K)	0.5297	0.6355	0.0888	0.7315	0.3422	0.3588	0.7433	0.8082
Trigr. BNC (10K)	**0.6289**	0.6053	0.4125	0.7961	0.4150	0.5468	0.7592	0.8677

5.2.1 Efficiency

In terms of efficiency, there is again a saving in space by reducing the size of the dataset. However, for this task the space reduction is less significant as it is a collection of a few megabytes. As stated before, our boilerplate removal method permits the utilisation of less expensive models, such as RoBERTa base which, together with the perplexity-based pre-processing, overcomes more complex models. This results in direct savings in training time. For instance, in our experiments the RoBERTa base model was more than 3 times faster than RoBERTa large.

On the other hand, perplexity-based pruning leads to smaller representations of the web-pages and, potentially, to lower prediction times. However, the state-of-the-art classification models described above, based on transformer technologies, do not analyse the entire input documents but are limited to a 512 token limit. In practice, even after removal of noisy sentences,

Table 7. CleanEval shared task results for two perplexity-based methods and the jusText algorithm. The percentage scores represent the average similarity between the webpages cleaned automatically and the ground truth webpages

	Effectiveness (%)
jusText	76.20
Bigrams model	
BNC (8K)	79.76
BNC (10K)	79.80

most documents are above the 512 token limit, and, thus, the improvement in prediction time was not noticeable. However, the decrease in prediction time would be observable under other classification models that make predictions based on the whole page.

5.3 Cleaning experiments: CleanEval

As stated before, our method goes one step further than simply removing boilerplate. Its main goal is to identify useful and well-formed text to enhance the performance of downstream tasks. However, we also wanted to demonstrate its validity for cleaning webpages. To this end, we selected the best performers for the previous tasks (both BNC-based bigrams models with thresholds set to 8k and 10k) and compared them with the jusText algorithm. As can be seen in Table 7, our models outperform jusText for the CleanEval shared-task.

6. Discussion

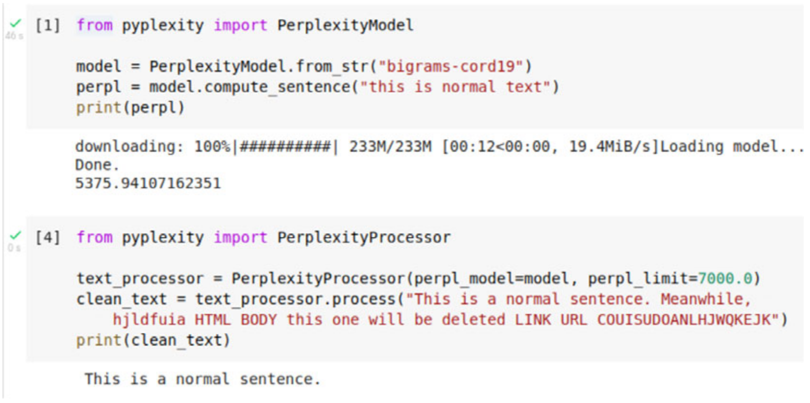
Our unsupervised cleaning method improved performance for most of the evaluation measures under the search task. However, it should be noticed that the perplexity-based removal works as a precision-oriented technique, and it was less effective in terms of recall (particularly in terms of AP). This occurs because our cleaning technique eliminates not only boilerplate, but also malformed or noisy sentences and some of them might be actually relevant.

Another important outcome is that for both tasks BNC-based models outperformed their CORD-19 counterparts. We wanted to test how the topicality of the background corpus influences performance. The results suggest that the most general model, in terms of content, yields the best results. This happened for both tasks (search and classification).

The proposed perplexity models consisted of simple bigrams and trigrams under a probabilistic approach. Experiments demonstrate that the bigram variants, which are less complex and thus more efficient, are also the top performers for most of the evaluated metrics. This saving was also one of the main objectives of the study, as stated at the beginning.

Therefore, not only effectiveness but also efficiency was taken into account in our study. As detailed in Section 5.1.3, our models substantially reduce carbon emissions, when a sufficient number of searches are performed over the collection (Lacoste *et al.* 2019). We estimate that our method could reduce in average 5.58kg of CO_2 per million queries. To put this result in perspective, this CO_2 weight is equivalent to 24.13 km driven by an average car[h] , 3.09 kg of coal burned[i] or 0.1 tree seedlings consuming carbon for 10 years[j] .

[h]https://www.epa.gov/energy/greenhouse-gases-equivalencies-calculator-calculations-and-references#miles
[i] https://www.epa.gov/energy/greenhouse-gases-equivalencies-calculator-calculations-and-references#lbscoal
[j] https://www.epa.gov/energy/greenhouse-gases-equivalencies-calculator-calculations-and-references#seedlings



```
[1] from pyplexity import PerplexityModel

    model = PerplexityModel.from_str("bigrams-cord19")
    perpl = model.compute_sentence("this is normal text")
    print(perpl)

    downloading: 100%|##########| 233M/233M [00:12<00:00, 19.4MiB/s]Loading model...
    Done.
    5375.94107162351

[4] from pyplexity import PerplexityProcessor

    text_processor = PerplexityProcessor(perpl_model=model, perpl_limit=7000.0)
    clean_text = text_processor.process("This is a normal sentence. Meanwhile,
        hjldfuia HTML BODY this one will be deleted LINK URL COUISUDOANLHJWQKEJK")
    print(clean_text)

     This is a normal sentence.
```

Figure 3. Some of the PYPLEXITY capabilities shown up and running in a Jupyter Notebook.

Finally, we have done a special effort to make the perplexity-based technology available to the community. We anticipate multiple uses of perplexity as a pre-processing mechanism in a wide range of text mining projects. To facilitate new applications of perplexity, we have created a Python package and a web demo that implement the boilerplate removal methods detailed in this study. This is described in the next section.

7. Python package and Web demo

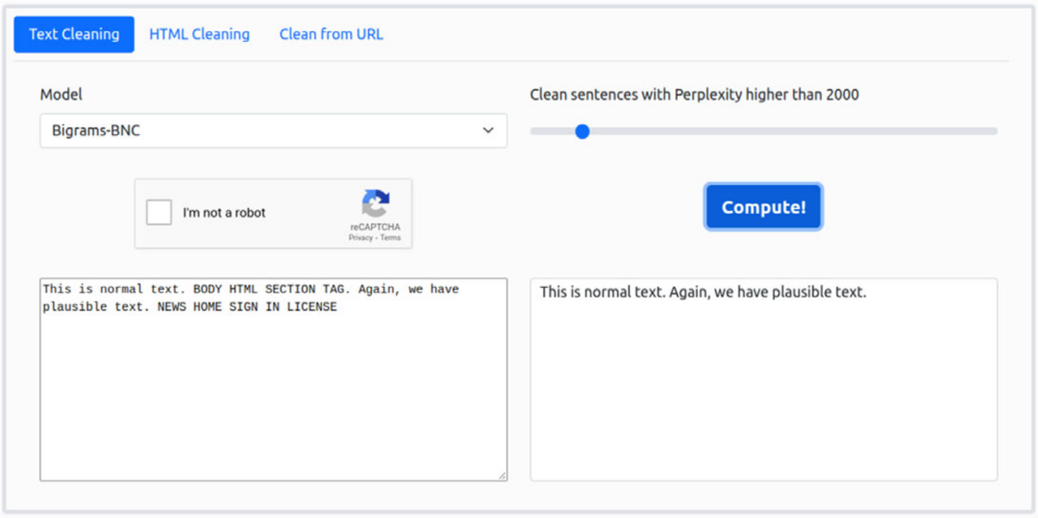
Along with this publication, we provide PyPlexity[a], a Python package available at the PyPi[k] repository. We also release with the package the source code required to train and utilise the models. This library serves two purposes: on one hand, it allows end-users to utilise our models from any python programme, with just a single line of code. On the other hand, it offers a command-line interface that provides straightforward perplexity computation.

There are three main commands that can be run from console. The simplest one is *perplexity*, which calculates the perplexity score for a sentence and a given model. The *tag-remover* command processes a raw HTML file or input directory of files and removes any tags and other non-text components. Finally, the *bulk-perplexity* processor computes perplexity for a batch of files and removes any sentence above the perplexity limit provided by the user (set by default to 8K). These two latter commands also have distributed computing capabilities, allowing efficient processing of large collections of documents in a computing cluster[l].

The Python interface allows integrating these computations into Python code. To demonstrate some of these capabilities, Figure 3 is provided, in which we utilise our tool to compute the perplexity score of a sentence and clean a line of text from Python code. Please refer to the documentation[m] for further details.

Additionally, we are publishing a web app[b] to make some small tests on how to directly clean raw and html texts or cleaning directly from an URL. A screenshot of the demo is shown in Figure 4.

[k]https://pypi.org/project/pyplexity/
[l]https://github.com/citiususc/pyplexity#parallel-mode-cluster
[m]https://github.com/citiususc/pyplexity#interfacing-from-python



Model: Bigrams-BNC

Clean sentences with Perplexity higher than 2000

```
This is normal text. BODY HTML SECTION TAG. Again, we have
plausible text. NEWS HOME SIGN IN LICENSE
```

```
This is normal text. Again, we have plausible text.
```

Figure 4. Demo webpage that showcases the capabilities of the methodology presented in this study.

8. Conclusion and future work

In this paper, we have proposed an unsupervised method for extracting useful content from scraped webpages. To the best of our knowledge, this is the first perplexity-based approach adopted for this kind of text pre-preprocessing.

A thorough experimentation process has been performed. This included evaluation with different test collections and tasks. With perplexity-based removal of sentences, we have managed to improve performance under several web-related tasks, in some cases even significantly.

In addition, our models also follow Green AI principles and seek to maximise energy efficiency. The pre-processing requirements of our method are compensated for the advantages associated with the resulting noise-free web contents when they are used in the test tasks.

Furthermore, we provide an easy-to-use library and web tool, which facilitate the adoption of this technology to support the removal of noisy textual extracts. We are convinced that perplexity-based removal might play a role in multiple application domains and text mining tasks.

As future work, we intend to explore other tokenisation techniques and work with different textual granularities (besides sentence level) to see the impact on recall-oriented measures. Moreover, we also plan to test the helpfulness of these cleaning techniques in other areas, such as health misinformation detection, and continue expanding the open-source tools which accompany this study.

Acknowledgments. The authors thank the support obtained from i) project PLEC2021-007662 (MCIN/AEI/10.13039/501100011033, Ministerio de Ciencia e Innovación, Agencia Estatal de Investigación, Plan de Recuperación, Transformación y Resiliencia, Unión Europea-Next GenerationEU), and ii) Consellería de Educación, Universidade e Formación Profesional (accreditation 2019-2022 ED431G-2019/04, ED431C 2022/19) and the European Regional Development Fund, which acknowledges the CiTIUS-Research Center in Intelligent Technologies of the University of Santiago de Compostela as a Research Center of the Galician University System.

Competing interests declaration. The author(s) declare none.

References

Abualsaud M., Lioma C., Maistro M., Smucker M.D. and Zuccon G. (2019). Overview of the TREC 2019 decision track. In *Proceedings of the 28th Text REtrieval Conference, (TREC '19)*, Gaithesburg, Maryland, USA. National Institute of Standards and Technology (NIST).

Baroni M., Chantree F., Kilgarriff A. and Sharoff S. (2008). Cleaneval: A competition for cleaning web pages. In *Lrec*, Marrakech.

Bauer D., Degen J., Deng X., Herger P., Gasthaus J., Giesbrecht E., Jansen L., Kalina C., Kräger T., Märtin R., *et al.* (2007). Fiasco: Filtering the internet by automatic subtree classification, osnabruck. In *Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop, Incorporating Cleaneval*, volume 4, pp. 111–121. Presses Univ. de Louvain.

Campos J.R.P., Otero P.G. and Loinaz I.A. (2020). Measuring diachronic language distance using perplexity: Application to English, Portuguese, and Spanish. *Natural Language Engineering* **26**, 433–454.

Collins-Thompson K., Bennett P.N., Diaz F., Clarke C. and Voorhees E.M. (2013). Overview of the TREC 2013 web track. In *Proceedings of the 22nd Text REtrieval Conference (TREC '13)*, Gaithesburg, Maryland, USA. National Institute of Standards and Technology (NIST).

Consortium B., *et al.* (2007). *British National Corpus*. Oxford, UK. University of Oxford.

Croft W.B., Metzler D. and Strohman T. (2010). *Search Engines: Information Retrieval in Practice*, volume **520**. Reading, Massachusetts, USA: Addison-Wesley Reading.

Csiszár I. (1975). I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, 146–158.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Duhart C., Dublon G., Mayton B., Davenport G. and Paradiso J.A. (2019). Deep learning for wildlife conservation and restoration efforts. In *36th International Conference on Machine Learning, Long Beach*, volume 5.

Finn A., Kushmerick N. and Smyth B. (2001). Fact or fiction: Content classification for digital libraries. In *DELOS*.

Gamallo P., Pichel J.R. and Alegria I. (2017). From language identification to language distance. *Physica A* **484**, 162–172.

González M. (2015). An analysis of twitter corpora and the differences between formal and colloquial tweets. In *Proceedings of the Tweet Translation Workshop 2015*, pp. 1–7, Alicante, Spain.

Henderson P., Hu J., Romoff J., Brunskill E., Jurafsky D. and Pineau J. (2020). Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research* **21**, 1–43.

Kannan S., Gurusamy V., Vijayarani S., Ilamathi J., Nithya M., Kannan S. and Gurusamy V. (2014). Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks* **5**, 7–16.

Kohlschütter C. (2009). A densitometric analysis of web template content. In *Proceedings of the 18th International Conference on World Wide Web*, Madrid, Spain, pp. 1165–1166.

Kohlschütter C., Fankhauser P. and Nejdl W. (2010). Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, New York, USA, pp. 441–450.

Lacoste A., Luccioni A., Schmidt V. and Dandres T. (2019). Quantifying the carbon emissions of machine learning. arXiv preprint arXiv:1910.09700.

Lee N., Bang Y., Madotto A. and Fung P. (2020). Misinformation has high perplexity. arXiv preprint arXiv:2006.04666.

Leonhardt J., Anand A. and Khosla M. (2020). Boilerplate removal using a neural sequence labeling model. In *Companion Proceedings of the Web Conference 2020*, Taipei, Taiwan, pp. 226–229.

Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L. and Stoyanov V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. CoRR, abs/1907.11692.

Losada D. and Azzopardi L. (2008). An analysis on document length retrieval trends in language modeling smoothing. *Information Retrieval* **11**, 109–138.

Moro A. and Lonza L. (2018). Electricity carbon intensity in european member states: Impacts on ghg emissions of electric vehicles. *Transportation Research Part D: Transport and Environment* **64**, 5–14.

Parapar J., Losada D.E. and Barreiro Á. (2021). Testing the tests: simulation of rankings to compare statistical significance tests in information retrieval evaluation. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pp. 655–664, Virtual Event Republic of Korea.

Parapar J., Losada D.E., Presedo-Quindimil M.A. and Barreiro A. (2020). Using score distributions to compare statistical significance tests for information retrieval evaluation. *Journal of the Association for Information Science and Technology* **71**, 98–113.

Pomikálek J. (2011). Removing boilerplate and duplicate content from web corpora. *Disertacn práce, Masarykova univerzita, Fakulta informatiky*.

Ponte J.M. and Croft W.B. (2017). A language modeling approach to information retrieval. In *ACM SIGIR Forum*, volume 51, pp. 202–208. New York, NY, USA: ACM.

Schwartz R., Dodge J., Smith N.A. and Etzioni O. (2020). Green AI. *Communications of the ACM* **63**, 54–63.

Sennrich R. (2012). Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, *EACL '12*, pp. 539–549, Association for Computational Linguistics: Stroudsburg, PA, USA.

Solorio T., Sherman M., Liu Y., Bedore L.M., Peña E.D. and Iglesias A. (2011). Analyzing language samples of spanish–english bilingual children for the automated prediction of language dominance. *Natural Language Engineering* **17**, 367–395.

Spousta M., Marek M. and Pecina P. (2008). Victor: the web-page cleaning tool. In 4th Web as Corpus Workshop (WAC4)-Can We Beat Google, pp. 12–17.

Sun A., Lim E.-P. and Ng W.-K. (2002). Web classification using support vector machine. In *Proceedings of the 4th International Workshop on Web Information and Data Management*, pp. 96–99, McLean Virginia, USA.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł. and Polosukhin I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, volume **30**.

Vieira K., Da Silva A.S., Pinto N., De Moura E.S., Cavalcanti J.M. and Freire J. (2006). A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, Arlington Virginia, USA, pp. 258–267.

Vogels T., Ganea O.-E. and Eickhoff C. (2018). Web2text: Deep structured boilerplate removal. In *European Conference on Information Retrieval*, pp. 167–179, Grenoble, France. Springer.

Wang L.L., Lo K., Chandrasekhar Y., Reas R., Yang J., Burdick D., Eide D., Funk K., Katsis Y., Kinney R.M., Li Y., Liu Z., Merrill W., Mooney P., Murdick D.A., Rishi D., Sheehan J., Shen Z., Stilson B., Wade A.D., Wang K., Wang N.X.R., Wilhelm C., Xie B., Raymond D.M., Weld D.S., Etzioni O. and Kohlmeier S. (2020). CORD-19: The COVID-19 open research dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online. Association for Computational Linguistics.

Wenzek G., Lachaux M.-A., Conneau A., Chaudhary V., Guzmán F., Joulin A. and Grave E. (2019). Ccnet: Extracting high quality monolingual datasets from web crawl data. arXiv preprint arXiv:1911.00359.

Wu C., Kanoulas E. and de Rijke M. (2020). It all starts with entities: A salient entity topic model. *Natural Language Engineering* **26**, 531–549.

Yang P., Fang H. and Lin J. (2018). Anserini: Reproducible ranking baselines using lucene. *Journal of Data and Information Quality (JDIQ)* **10**, 1–20.

Yi L., Liu B. and Li X. (2003). Eliminating noisy information in web pages for data mining. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington D.C., USA, pp. 296–305.

Cite this article: Fernández-Pichel M, Prada-Corral M, Losada DE, Pichel JC and Gamallo P (2024). An unsupervised perplexity-based method for boilerplate removal. *Natural Language Engineering* **30**, 132–149. https://doi.org/10.1017/S1351324923000049